THE OPTIMISATION OF LARGE SCALE LOGICAL CIRCUITS

Pavel Seda
Doctoral Degree Programme (1), FEEC BUT
E-mail: xsedap01@feec.vutbr.cz

Supervised by: Jiri Hosek

Abstract: In the phase of designing the logical circuits, it is essential to minimise the number of elements because it leads to the more reliable, more secure, and cheaper solution. For the logical functions with less than 4 variables, the Karnaugh maps are suitable. However, in practice, we encounter usually a much more complex function, in those cases, we could apply Boolean algebra laws directly or use the Quine-McCluskey method, which is based on their systematic use. Unfortunately, this method does not usually provide a minimal form of logical function for really large scale logical functions, and in a result may be redundant expressions. For that reason, we show that we could apply an additional phase which leads to the set covering problem which needs to cover all the inputs by the obtained outputs. Since this problem is $\mathcal{NP}$-hard, it is necessary to use heuristic methods, such as simulated annealing.

Keywords: logic circuits, minimisation, set covering problem, simulated annealing

1 INTRODUCTION

It is known from the uniting theorem of Boolean algebra that a disjunction of two logical conjunctions with the same variables and differing in one of them in a complementary occurrence can be simplified to their common part. Here, distributive law, complement law and neutrality law were applied.

For example $\bar{a}\bar{b}cd + \bar{a}\bar{b}\bar{c}d = \bar{a}\bar{b}d(c+\bar{c}) = \bar{a}\bar{b}d \cdot 1 = \bar{a}\bar{b}d$.

For functions with a greater number of expressions, we can also use the idempotence law $x+x=x$, and therefore to use any expression as many times as it enables to simplify one of the other expressions. This is used in Karnaugh maps when some of the expressions are assigned to several groups.

Example 1. Minimise the following logical function

$$f = \bar{a}\bar{b} + abd + (\bar{b}cd \text{ or } acd).$$

Denote the conjunctions in f as follows:

$$\begin{array}{lll} 1 \to \bar{a}\bar{b}\bar{c}\bar{d} & 4 \to \bar{a}\bar{b}cd & 6 \to abcd \\ 2 \to \bar{a}\bar{b}cd & 5 \to a\bar{b}cd & 7 \to ab\bar{c}d \\ 3 \to \bar{a}\bar{b}\bar{c}d & & \end{array} \tag{1}$$

Since we have only four variables, we can easily solve the example using the Karnaugh map [1]. By the previous denotations, expressions are placed in positions as in Fig. 1.

These expressions are represented in the Karnaugh map by one.

If we denote groups of neighbouring expressions in Fig. 2 and simplify them in common parts, we get the following minimum form of the f function:

$$f_{1min} = \bar{a}\bar{b} + abc + \bar{b}cd. \tag{2}$$

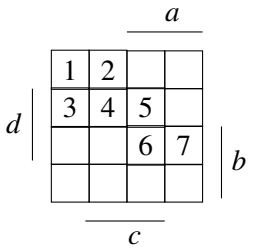

Figure 1: Positions of expressions.

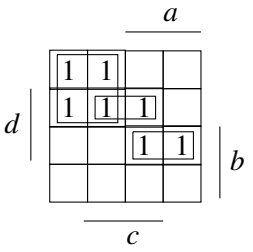

Figure 2: Karnaugh map.

The third expression in Eq. (2) was obtained by grouping 1 in the 2nd row and the 3rd column with the left-handed neighbour.

However, we can group this 1 with the lower neighbour and get another, just as good solution by Eq. (3).

$$f_{2min} = \bar{a}\bar{b} + abc + acd. \tag{3}$$

△

1.1 QUINE-MCCLUSKEY METHOD

As already mentioned, this graphical approach is only applicable to logic functions with 2, 3, or 4 variables, and for a larger number of variables it is necessary to use, e.g., the Quine-McCluskey method.

Its principle is as follows:

1. Conjunctions of a given logical function are divided into groups such that expressions with the same number of negations are in the same group.
2. All pairs of conjunctions from neighbouring groups (i.e. groups whose number of negations differ by one) are compared.
3. If the uniting theorem may be applied, then these expressions are removed and a common part is carried to the next iteration.
4. If the number of expressions in the new iteration is nonzero, then steps 2-4 are repeated, otherwise the algorithm finishes [2].

Now, we will apply this method to the same example and compare the results. Expressions, that will be replaced by a simpler form, will be written in parentheses. The final result will then be given by the disjunction of all expressions without parentheses.

(0)

$abcd$
$a\bar{b}cd, abc\bar{d}$
$\bar{a}\bar{b}cd$
$\bar{a}\bar{b}cd, \bar{a}\bar{b}\bar{c}d$
$\bar{a}\bar{b}\bar{c}\bar{d}$

(1)

acd, abc
$\bar{b}cd$
$(\bar{a}\bar{b}c), (\bar{a}\bar{b}d)$
$(\bar{a}\bar{b}\bar{d}), (\bar{a}\bar{b}\bar{c})$

(2)

acd, abc
$\bar{b}cd$
$(\bar{a}\bar{b}c), (\bar{a}\bar{b}d)$
$(\bar{a}\bar{b}\bar{d}), (\bar{a}\bar{b}\bar{c})$

Summarising all the expressions that have not been replaced by simpler ones, we get the following result:

$$f_{3min} = acd + abd + \bar{b}cd + \bar{a}\bar{b}. \tag{4}$$

Obviously, the expressions in the result f_{3min} are the union of the expressions in f_{1min} and f_{2min}, and thus f_{3min} is not minimal.

Since the output expressions in the result f_{3min} are derived from the input expressions of the given logical function f, only such a set of output expressions must be left to provide a link to all inputs.

The output has a connection to the input if it is a part of it, i. e. the output was simplified from the input. In Table I these connections are specified.

Table 1: Quine-McCluskey method

outputs	input1 $\bar{a}\bar{b}\bar{c}\bar{d}$	input2 $\bar{a}\bar{b}cd$	input3 $\bar{a}\bar{b}\bar{c}d$	input4 $\bar{a}\bar{b}cd$	input5 $a\bar{b}cd$	input6 $abcd$	input7 $ab\bar{c}d$
$\bar{a}\bar{b}$	1	1	1	1	0	0	0
$\bar{b}cd$	0	0	0	1	1	0	0
acd	0	0	0	0	1	1	0
abc	0	0	0	0	0	1	1

Since columns 1, 2, 3 and 7 contain value 1 only once, the corresponding outputs in lines cannot be omitted. That means the 1st and the 4th output must be in the result. These two outputs are linked to all inputs except for the 5th input. Therefore, the 2nd or the 3rd output must be added to complete the cover.

2 RELATED WORK

In the previous sections, we have shown that the minimisation of logical functions using the Quine-McCluskey method requires a post-processing phase based on solving the Set Covering Problem (SCP).

This problem has a wide area of applications, e.g., minimisation of service centres networks (schools, hospitals, financial offices, ...) to guarantee that each user of this service has at least one centre in a reachable distance, which is defined by a threshold value.

But what is substantial, is the fact that the problem has a complexity of $O(2^m)$, which expresses the number of possible combinations of m outputs among them we select to complete the cover. For large instances it needs heuristic methods. In [3] a new modification of genetic algorithm was proposed, which improves [4]. However, we also need the SCP model, because it is important for the problem representation.

3 MINIMISATION USING SET COVERING PROBLEM AND MEASUREMENTS

3.1 MODEL FOR SET COVERING PROBLEM

The corresponding mathematical model is as follows [3, 5]:

Minimise

$$z = \sum_{i=1}^{m} w_i x_i \tag{5}$$

subject to

$$\sum_{i=1}^{m} a_{ij} x_i \geq 1, j = 1, \ldots, n \tag{6}$$

$$x_i \in \{0,1\}, i = 1, \ldots, m. \tag{7}$$

x_i is the output in the logical circuits, if x_i=1 then the output is in the solution and if otherwise then not. $a_{ij} = 1$ means that output x_i covers input j. w_i represents the weights of logical expressions, e.g., the number of negations. Due to minimisation, the smaller these numbers are, the smaller the coefficients must be. From this point of view f_{2min} would be better than f_{1min}.

Here, we use the simulated annealing (SA) technique, which is based on simulating the cooling of a material in a heat bath – a process known as annealing. Outputs, among them is necessary to select, are represented by a binary string. To avoid infeasible solutions when a neighbour is generated, a repair operator must be applied.

Algorithm 1 Repair Operator in SCP for minimisation of output set to reach a valid solution.

Input: $OUT = \{1, \ldots, m\}$ = the set of all outputs (rows); $IN = \{1, \ldots, n\}$ = the set of all inputs (columns); $OUTS$ = the set of outputs in a solution; UIN = the set of uncovered inputs; w_i = the number of outputs that cover input $j, j \in IN$ in OUT; $\alpha_j\{i \in OUT \mid a_{ij} = 1\}$ = the set of outputs that cover input $j, j \in IN$; $\beta_i = \{j \in IN | a_{ij} = 1\}$ = the set of inputs that are covered by outputs i, $i \in OUT$;

```
    procedure REPAIROUTPUTSET()
        initialize w_j = |OUT ∩ α_j|, ∀j ∈ IN;
        initialize UIN = {j|w_j = 0, ∀j ∈ IN};
 4:     for all inputs j in UIN do
            find the first output i in α_j that minimises
            1 / |UIN ∩ β_i|;
            OUTS ← OUTS + i;
 8:         w_j ← w_j + 1, ∀j ∈ β_i;
            UIN ← UIN − β_i;
        end for
        for all output i in OUTS do
12:         if w_j ≥ 2, ∀_j ∈ β_i then
                OUTS ← OUTS − i;
                w_j ← w_j − 1, ∀_j ∈ β_i;
            end if
16:     end for
    end procedure        ▷ {The set of outputs is now a feasible solution without redundant outputs}
```

3.2 SIMULATIONS

In this paper, the simulated were provided against benchmark datasets from OR-Library [6]. The algorithm of SA was chosen to minimise the given matrices of inputs and outputs (representing the logical circuit).

The parameters for SA were set as follows: (i) init temperature 10000.0, (ii) final temperature 1, and (iii) cooling rate of 0.999.

The two benchmark datasets were used from OR-Library, the first was initially described by 23 elements inside and thanks to our optimisation it was reduced to 6 logical elements. The process of minimisation using the SA algorithm is plotted in Fig. 3.

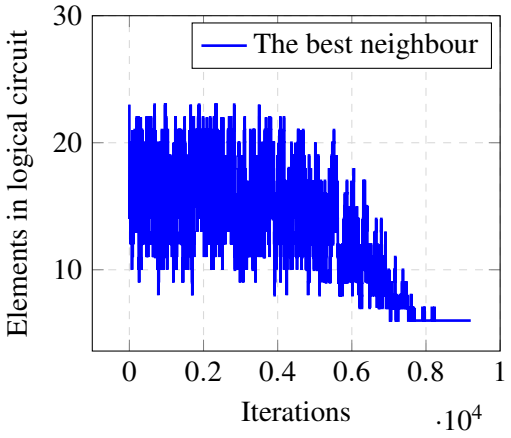

Figure 3: The logical circuit with initially 23 elements reduced to 6.

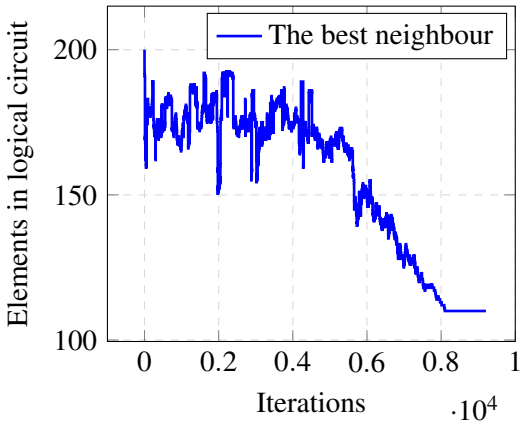

Figure 4: The logical circuit with initially 200 elements reduced to 110.

The second simulation was against dataset with 200 outputs and 2000 inputs, which was minimised to 110 outputs. It is shown in Fig. 4.

4 CONCLUSION

In this paper, we introduced a model for the minimisation of logical functions. We showed that the general version of the problem with many variables solved by the Quine-McCluskey method needed a post-processing procedure. This second phase computation corresponds to the SCP problem, which is $\mathcal{NP}$-hard with a complexity of $O(2^m)$ and, thus, in order to solve larger instances, arbitrary meta-heuristics must be used. We propose a simulated annealing technique based on binary string representation and a repair operator. The model is validated on a standard OR-library benchmark dataset and with presented parameter settings give sub-optimal in very short times.

REFERENCES

[1] G. Epstein, *Multiple-valued logic design: an introduction*. Routledge, 2017.

[2] X. Huang, N. Wu, and X. Zhang, "Quine-mccluskey repair technique for evolutionary design of combinational logic circuits," in *Proc. International MultiConference of Engineers and Computer Scientists 2015*, pp. 674–678, 2015.

[3] P. Seda, M. Mark, K. Su, M. Seda, J. Hosek, and J. Leu, "The minimization of public facilities with enhanced genetic algorithms using war elimination," *IEEE Access*, vol. 7, pp. 9395–9405, 2019.

[4] J. E. Beasley and P. C. Chu, "A genetic algorithm for the set covering problem," *European journal of operational research*, vol. 94, no. 2, pp. 392–404, 1996.

[5] J. M. Lanza-Gutierrez, B. Crawford, R. Soto, N. Berrios, J. A. Gomez-Pulido, and F. Paredes, "Analyzing the effects of binarization techniques when solving the set covering problem through swarm optimization," *Expert Systems with Applications*, vol. 70, pp. 67–82, 2017.

[6] J. E. Beasley, "Or-library: distributing test problems by electronic mail," *Journal of the operational research society*, vol. 41, no. 11, pp. 1069–1072, 1990.